UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/871,475	05/31/2001	Reto Preisig	SVL920010020US1	1554

7590 03/15/2004

John L. Rogitz
Rogitz & Associates
750 B Street, Suite 3120
San Diego, CA 92101

EXAMINER	
LE, DEBBIE M	

ART UNIT	PAPER NUMBER
2177	9

DATE MAILED: 03/15/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Advisory Action	Application No.	Applicant(s)	
	09/871,475	PREISIG ET AL.	
	Examiner	Art Unit	
	DEBBIE M LE	2177	

--The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

THE REPLY FILED 27 February 2004 FAILS TO PLACE THIS APPLICATION IN CONDITION FOR ALLOWANCE. Therefore, further action by the applicant is required to avoid abandonment of this application. A proper reply to a final rejection under 37 CFR 1.113 may only be either: (1) a timely filed amendment which places the application in condition for allowance; (2) a timely filed Notice of Appeal (with appeal fee); or (3) a timely filed Request for Continued Examination (RCE) in compliance with 37 CFR 1.114.

PERIOD FOR REPLY [check either a) or b)]

a) ☒ The period for reply expires 3 months from the mailing date of the final rejection.

b) ☐ The period for reply expires on: (1) the mailing date of this Advisory Action, or (2) the date set forth in the final rejection, whichever is later. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of the final rejection.
ONLY CHECK THIS BOX WHEN THE FIRST REPLY WAS FILED WITHIN TWO MONTHS OF THE FINAL REJECTION. See MPEP 706.07(f).

Extensions of time may be obtained under 37 CFR 1.136(a). The date on which the petition under 37 CFR 1.136(a) and the appropriate extension fee have been filed is the date for purposes of determining the period of extension and the corresponding amount of the fee. The appropriate extension fee under 37 CFR 1.17(a) is calculated from: (1) the expiration date of the shortened statutory period for reply originally set in the final Office action; or (2) as set forth in (b) above, if checked. Any reply received by the Office later than three months after the mailing date of the final rejection, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

1. ☐ A Notice of Appeal was filed on _____. Appellant's Brief must be filed within the period set forth in 37 CFR 1.192(a), or any extension thereof (37 CFR 1.191(d)), to avoid dismissal of the appeal.

2. ☐ The proposed amendment(s) will not be entered because:

   (a) ☒ they raise new issues that would require further consideration and/or search (see NOTE below);

   (b) ☐ they raise the issue of new matter (see Note below);

   (c) ☐ they are not deemed to place the application in better form for appeal by materially reducing or simplifying the issues for appeal; and/or

   (d) ☐ they present additional claims without canceling a corresponding number of finally rejected claims.

   NOTE: *See Continuation Sheet*.

3. ☐ Applicant's reply has overcome the following rejection(s): _____.

4. ☐ Newly proposed or amended claim(s) _____ would be allowable if submitted in a separate, timely filed amendment canceling the non-allowable claim(s).

5. ☐ The a)☐ affidavit, b)☐ exhibit, or c)☐ request for reconsideration has been considered but does NOT place the application in condition for allowance because: _____.

6. ☐ The affidavit or exhibit will NOT be considered because it is not directed SOLELY to issues which were newly raised by the Examiner in the final rejection.

7. ☒ For purposes of Appeal, the proposed amendment(s) a)☒ will not be entered or b)☐ will be entered and an explanation of how the new or amended claims would be rejected is provided below or appended.

   The status of the claim(s) is (or will be) as follows:

   Claim(s) allowed: _____.

   Claim(s) objected to: _____.

   Claim(s) rejected: *1-32*.

   Claim(s) withdrawn from consideration: _____.

8. ☐ The drawing correction filed on _____ is a)☐ approved or b)☐ disapproved by the Examiner.

9. ☐ Note the attached Information Disclosure Statement(s)(PTO-1449) Paper No(s). _____.

10. ☐ Other: _____

NCC

3/12/04

John E. Breene

JOHN BREENE
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Continuation of 2. NOTE: The Declaration filed on Feb. 27, 2004 (pp# 8) requires more than cursory review. The examiner respectfully submits that the provisional application No. 60/214,067 fully supports the present invention claims rejection. A copy of the provisional application is provided to the applicants attached herein..

3/12/04

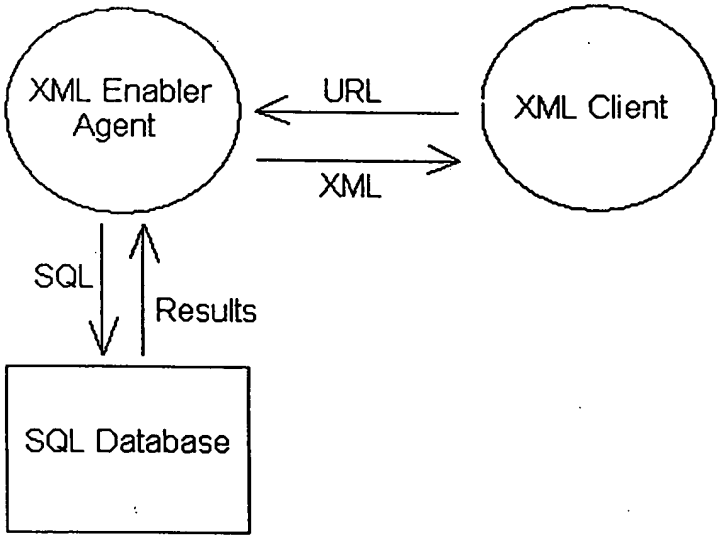
Emily XML Enabler Agent
Feature Specification

This document provides a specification of operation and for the Vertical Computer Systems Emily XML Enabler Agent. The document is intended to be an engineering specification for review purposes. The contents of this document may also be applicable to the generation of marketing materials and sales support information.

1. Item Description

The Emily XML Enabler Agent is a web-based application that co-exists with an SQL database, and creates XML formatted documents in response to SQL queries. This allows XML clients (such as BizTalk, the Emily Framework scripting language, and other client programs) to extract data from a database in a standard, platform independent way.

A top-level diagram of the XML Enabler Agent is provided below:



As shown in the above depiction, the XML Enabler Agent retrieves data for an XML client from an SQL database. The XML client accesses the SQL data by opening a URL. Additionally, the XML Enabler Agent can format data as HTML output (to provide general utility and assist in debug.)

2. Discussion

The XML agent is a light-weight server program that runs out-of-the-box. It can be easily configured to make relational data accessible to users via XML. The agent leverages existing specifications of XML, and provides the new concept of allowing queries via HTTP "post" commands.

XML client programs can access the XML agent via a variety of techniques. Additionally, human operators can access data in HTML format, and make changes to the system via a web browser. This dual mode of operation makes the agent useful, easy to implement, and easy to demonstrate and understand.

The XML Enabler Agent has application in e-commerce by presenting catalogs of items to electronic buying programs. The XML Enabler Agent administrator can compose queries in a format suitable for the buying program, and then make the URL to this query available to the buying program, such as by posting the URL in a form at the buyer's web site.

The XML Enabler Agent has application in data management by allowing configuration items to be easily fetched and parsed with programs and server processes. For example, a program can post to the XML Enabler Agent to fetch operating parameters of a system, or a list of configuration items that control the performance of a parameter.

Finally, the XML Enabler Agent provides general utility in debugging and viewing relational databases, such as locating records or filtering data. This permits interoperability with existing XML and HTML clients, in particular the Emily Framework. Additionally, this furnishes a web-based human interface to this data.

3. Feature Overview

The XML Enabler Agent provides both XML output and HTML output. The XML output is the main program interface, while the HTML output provides a web-based user interface.

- **XML Output.** The user can acquire XML output by posting an SQL statement to the agent, or by accessing a special URL configured by the administrator to generate a pre-defined query. This is the main program interface to the XML Enabler Agent, discussed in section 3.1.

- **HTML Output.** The user can acquire HTML output (for debug purposes or for exploring the capabilities of the agent) by accessing a top level URL, and then operating the web-based system. This provides a graphical, web-based user interface to the XML Agent, discussed in section 3.2.

These features are discussed in more detail within the next sections.

3.1 Agent XML Output

The primary interface of the agent is via XML. The user accesses data from the database through HTTP posts of SQL statements, or by accessing special URLs configured by the agent administrator. The output of the agent is an XML document, delivered via HTTP.

To use the XML interface requires and XML client, such as the Emily Framework, BizTalk, or cascading style sheets. To facilitate checkout and debug, the Agent incorporates a simple, web-based "tag browser" as a standard feature. (See section 3.2.)

3.1.1 Output Data Format

By default, the specific format of the XML data that is output by the XML Enabler agent is as follows:

```
<!?xml>
<query url=(url)/>

<record number=1>
   <field1>Value 1</field1>
   <field2>Value 2</field2>
   <fieldn>Value N</fieldn>
</record>
```

By default, the "field1", "field2", etc. tags are given the names of the table columns matched by the specified query. The "<record>" container tags are repeated for each table row returned by a particular query, to a user configurable limit. The user can map different names to the tag values through web-based setup screens.

By default, no DTD is generated for the XML document. The user can create a DTD that allows the particular types of each field to be specified, with a default value being PCDATA (i.e. character string data.) Using this technique, a valid and well-formed XML document can be constructed directly from a combination of the field names of the relation database table, and configuration data setup by the administrator.

3.1.2 Accessing XML Data

The user can access XML documents two different ways, each of which can be configured by the administrator to be the default (or only) method of accessing data. This provides alternate methods of acquiring XML based upon the specific requirements of the organization and XML client programs.

The following two methods of accessing XML data are provided.

- **HTTP URLs.** This is the simpler (and more secure) way of accessing XML data, but requires that the agent be setup by the administrator. Specifically, the user can access XML data via the following URL:

  ```
  http://(hostspec)/mle-cgi/xml?(Queryname)
  ```

  Above, the "Queryname" value is configured by the administrator with the web-based interface. This is an arbitrary identifier or keyword that is mapped to an SQL query on the system.

- **HTTP Post Commands.** The user can access the XML data by posting an SQL query to the system, and accessing the resulting document. This method relies on an HTTP "post" command to get the data, which simplifies the setup of the agent, but complicates the logic of the XML client (which has to compose the post request.) The URL to access this is as follows:

  ```
  http://(hostspec)/mle-cgi/xml?post
  ```

  When using the above technique, the "servername", "username", "password", and "query" arguments are provided via CGI. All CGI arguments (with the exception of the query) are optional and default to values configured by the administrator via the web-based interface.

The administrator can allow (or disallow) either or both of these options based upon the configuration of trusted hosts. (See section 4.1.)

The above techniques allow flexibility in querying XML data. The first technique requires the administrator to setup queries, which are then accessed by keyword. (This provides secure access to XML data, since the administrator can limit the range of options.) The second technique can be used "out-of-the-box", but can be used only with trusted clients, since allowing clients to compose queries will provide full visibility into the database.

3.2 HTML Output

To facilitate the setup, operation, and debug of the system, the XML Enabler provides a suite of CGI scripts that serve as a web-based graphical user interface. These screens provide general utility in setting up the agent, and debugging the agent.

After installation, operators can configure the agent using HTML based screens. The following basic screen set is provided as a standard feature of the XML Enabler Agent system.

- **Query Viewer Screen.** This screen is depicted in figure 1, and allows the user to view the contents of a query in HTML format. This screen allows an operator to view the query contents in human readable form.

- **Query Editor.** This screen is depicted in figure 2, and allows the user to construct queries that are associated with specific URLs. This screen provides general utility in creating queries that are made by the agent.

- **SQL Parameter Editor.** This screen is depicted in figure 3, and allows the user to configure the basic login parameters of the agent, such as the server name, login name, password, and potentially other parameters.

- **Security Parameters.** This screen allows the user to configure security parameters of the agent, including the list of trusted hosts, the authentication mechanism, and the operating modes of the system.

- **Tag Browser Utility.** This provides a simple web-based XML client that can be used to test the operation of the XML Enabler Agent.

Each of these screens is described below.

3.2.1 Query Viewer Screen

The SQL Viewer Screen allows the user to select a particular query to be viewed in HTML format. The screen shows the queried data in tabular format, and allows the user to select the particular query via a pull down menu. The various queries available to the user are provided in a pull down menu. SQL queries are constructed via the "Query Editor" tool shown in figure 2.

The Query Viewer screen provides a simple way to obtain visibility into a system for debug and analysis purposes. The table displayed by this screen is automatically created from the results of the query. The top of the table indicates the various fields (specified in the query), which also corresponds to the various XML tags read by the XML client.

The Query Viewer Screen, in addition to providing a view of queried information, also provides the user with a list of the various queries (created by the Query Editor Screen) from the pull down menu.

3.2.2 Query Editor Screen

The Query Editor Screen allows the user to compose queries that are associated with particular items in the pull down menu of the Query Viewer Screen (described above). This screen is depicted in figure 2. The Query editor screen allows the user to create, modify, or delete queries, and associate these queries with items in the pull down menu of the Query Viewer screen.

The Query Editor screen facilitates associations of SQL Queries (which are not visible to XML clients or unprivileged users) and the query results. Each query resides in its own file on the system that can be modified by the Query Editor, or other text file editor.

3.2.3 SQL Parameter Editor Screen

The SQL Parameter Editor screen provides general utility in configuring the various parameters of the system. This screen provides a central place for specifying values needed to make queries, such as driver programs, time-outs, passwords, and security items. This screen will normally be available only to privileged users. An example of this screen is depicted in figure 3.

3.2.4 Security Editor Screen

The Security Editor screen allows the user to configure the security for the program, as discussed in section 4.1. Specifically, the screen allows a list of trusted IP addresses to be established that can make queries, or access XML data.

3.2.5 Tag Browser Screen

The Tag Browser screen provides general utility in viewing the XML (or HTML) composition of a URL. This utility adds value to the system by viewing the XML output of the agent in a fashion similar to other browsers. The screen allows the operator to specify a URL, and view the various tag components of the resulting HTTP document.

4. Other Features

The XML Enabler Agent is intended to be comprehensive system containing various support tools and facilities:

- **Embedded HTTP Server.** The XML Enable agent comes with a ready-to-run HTTP server. Users can implement this HTTP server, or can install the XML Enabler Agent under an existing HTTP server.

- **Administrative / Security Console.** The XML Enabler Agent comes with a security console that allows administrators to set passwords and permissions on the various screens, and also on the embedded HTTP server. This console is a windows application that can be locked down by the administrator.

- **SQL Insert and Update Utilities.** The XML Enabler Agent comes with SQL Insert and Update utilities, that allow users (given suitable permissions by the administrator) to insert and update tables. This also provides a programmatic interface to allow a table to be loaded by the Emily Framework scripting language, or HTTP client.

- **Extensible API.** The XML Enabler Agent comes with user modifiable scripts that allow the screens and functions to be tailored by administrators. Additionally, the Emily Framework scripting language and development kit is optionally available to allow users to create new function and capabilities for the system.

4.1 Security Features

The XML Enabler Agent system provides several security features to limit usage by authorized personnel. These security features are as follows:

- **Secure Login Interface.** The XML Enabler Agent permits an administrator to use HTTP authentication, securing any screen or directory with a pop-up login interface. Logins to the system are configured via the web interface, or via a security console program.

- **Trusted Host List**. The XML Enabler Agent allows an operator to configure a list of trusted host names (by IP address or by network) that are permitted to access files and screens. This list of hosts specifies the method by which XML data can be acquired (such as by HTTP post, URL, or both). Additionally, this list of hosts specifies who can access the administrative HTML screens of the agent system.

- **Data Encryption via SSL.** Because the XML Enabler Agent is HTTP based, users can encrypt data via SSL. Although the native Apache Server that comes with the agent is not secure, this HTTP agent can be easily substituted for an SLL server with no loss of functionality.

4.2 Online Documentation.

The XML Enabler Agent incorporates online help in the form of hypertext help-files and PDF files. This documentation is sufficient to install, configure, operate, and maintain the system. Additional documentation may be available (as an option) describing how to extend the XML Enabler Agent.

5. Attachments

The remaining pages of this document provide examples of the HTML web-based interface, as discussed in section 3.2. These screens are provided only to show the general format of the XML Agent screens, and are not intended to be definitive illustrations of the agent's operation, or a representation of the actual delivered product.

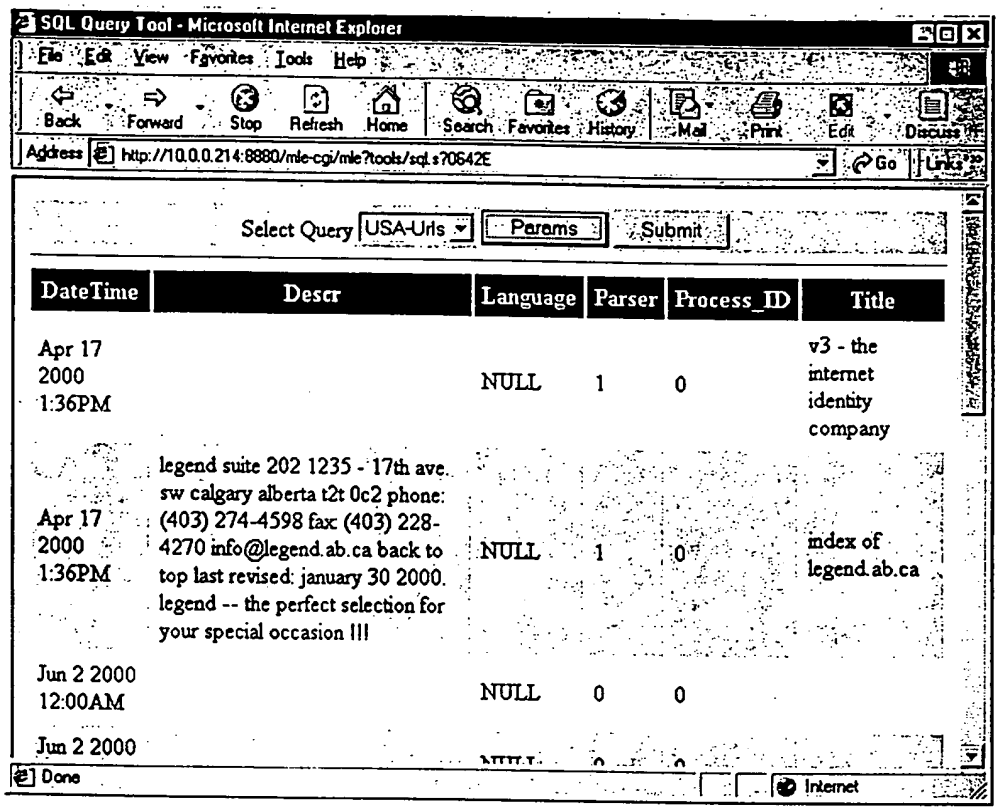


Figure 1: SQL Viewer Screen.

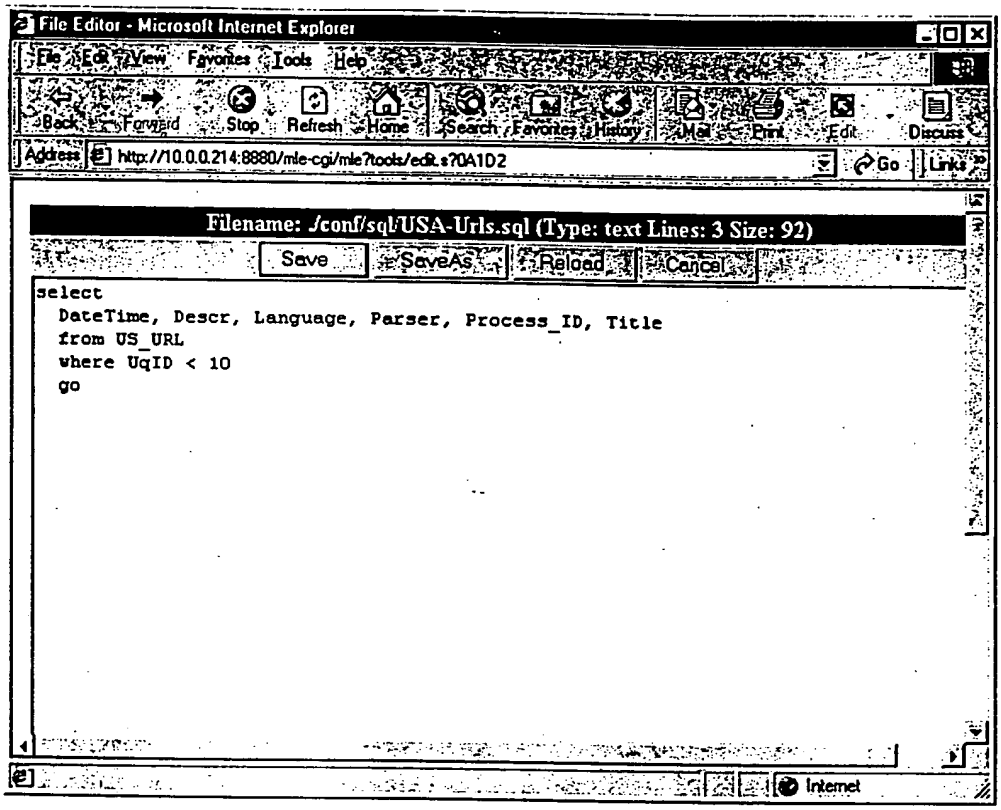


Figure 2: Query Editor Screen.

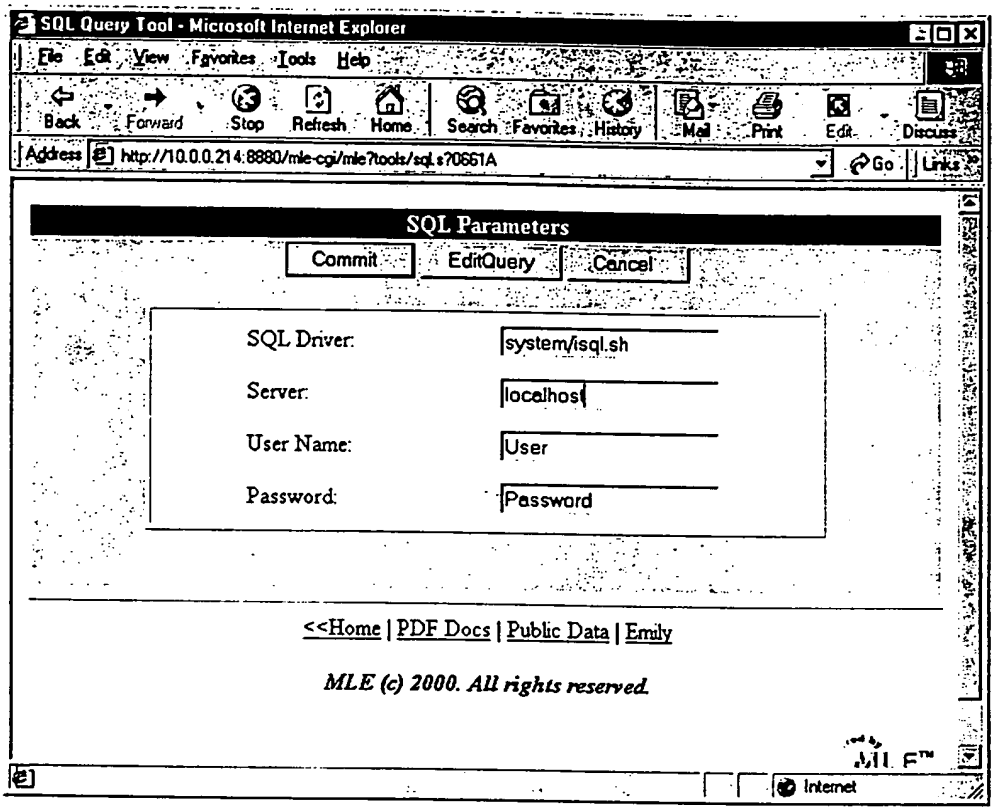


Figure 3: Agent Parameter Editor Screen

DESCRIPTION OF EMILY FRAMEWORK SCRIPTING LANGUAGE

With reference to Fig. 4, a flow diagram illustrating a typical method performed by the optional Emily scripting language is shown. The Emily scripting language may be used for processing a markup language file having one or more tagged portions. The method comprises opening a first markup language file, step 300, and parsing the first markup language file for one or more portions, step 310. The language interpreter then stores each portion of the first markup language file into one or more objects in an electronic memory, step 302.

Part of the Emily language may include a CAT or DIR command. If such a command is received by the language interpreter, step 306, then the interpreter causes the one or more objects to be presented for selection, viewing or other processing in one more corresponding folders and sub-folders on-screen, step 308. The one or more folders may be presented in a hierarchical list having sub-folders according to an arrangement of sub-portions of the first markup language file, each sub-folder representing an arrangement, sub-arrangement, or object containing a portion of the first markup language file.

The Emily language comprises a command language set allowing selection, viewing and other processing of the one or more objects. The command language set comprises a plurality of commands for selection, viewing and other processing. A subset of commands may comprise one or more commands for processing one or more folders, subfolders, portions, or subportions of the first markup language file, the subset comprising one or more executable batch files containing a subset of the set of commands. One or more executable batch files may be included within a second markup language file, the subset of commands in the executable batch file comprising commands for including one or more of the objects containing portions of the first markup language file in the second markup language file.

The language interpreter may receive one or more SET or UPDATE commands for setting or update objects in the first or second markup language file, step 310. If such a command is received, the language interpreter updates the objects in memory according to the commands received, step 312.

If a POST or SAVE command is received, step 314, the following steps may be performed as part of said other processing for updating the respective markup language file, step 316: receiving a subset of commands comprising one or more commands, said one or

more commands comprising instructions for updating one or more objects based on the received one or more commands; updating the one or more portions contained in the one or more objects according to the received one or more commands; and saving the portions contained within the one or more objects to the first respective language file that have been updated.

Alternatively, the following steps may be performed as part of said other processing for providing a second markup language file to a network node identified by a uniform resource locator: receiving a subset of commands comprising one or more commands, said one or more commands comprising instructions for updating one or more objects based on the received one or more commands; updating the one or more portions contained in the one or more objects according to the received one or more commands; and providing the second markup language file to the network node identified by the uniform resource locator.

Alternatively, the first markup language file may comprise one or more portions receptive to data input, wherein said other processing comprises receiving and sending said data input by performing the steps of: receiving said data input, storing said data input into in said objects containing said one or more portions receptive to data input; and posting said data input to said markup language file. At least a portion of the data input may then be processed after the step of receiving and before the step of storing. This alternative processing may be used, for example, to receive SQL commands for performing a query on a remote database, or posting to a remote database.

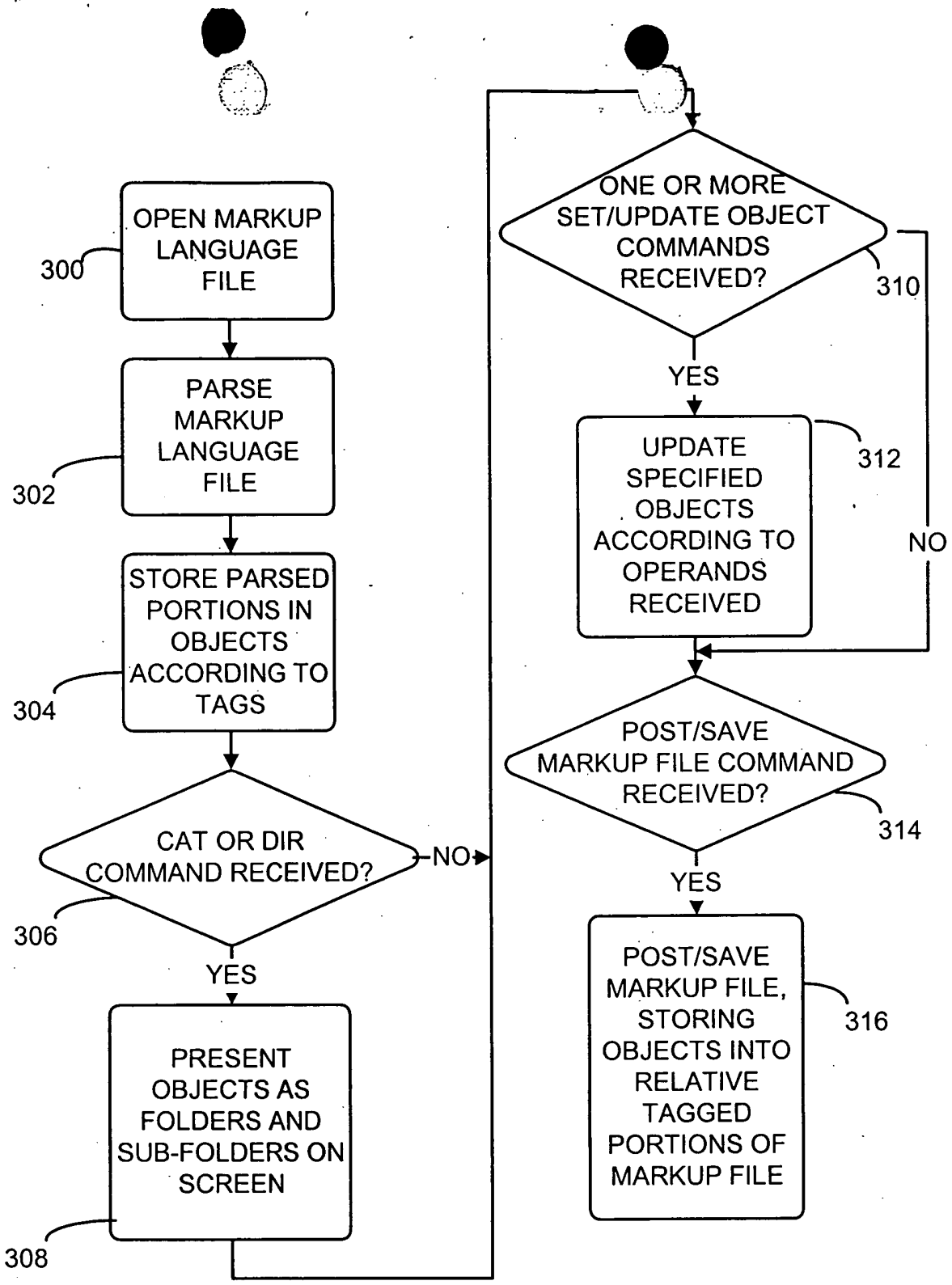

OPEN MARKUP LANGUAGE FILE
300
PARSE MARKUP LANGUAGE FILE
302
STORE PARSED PORTIONS IN OBJECTS ACCORDING TO TAGS
304
CAT OR DIR COMMAND RECEIVED?
306
NO
YES
PRESENT OBJECTS AS FOLDERS AND SUB-FOLDERS ON SCREEN
308
ONE OR MORE SET/UPDATE OBJECT COMMANDS RECEIVED?
310
YES
UPDATE SPECIFIED OBJECTS ACCORDING TO OPERANDS RECEIVED
312
NO
POST/SAVE MARKUP FILE COMMAND RECEIVED?
314
YES
POST/SAVE MARKUP FILE, STORING OBJECTS INTO RELATIVE TAGGED PORTIONS OF MARKUP FILE
316

Fig. 4